Implementing WIPI for Linux-based Smartphone

Jaeho Lee, Sunja Kim, Sangyun Lee, Woosik Kim, Hwangu Lee
Wireless Internet Platform Team, Embedded S/W Research Division, ETRI
{bigleap, sunjakim, sylee, wsk, hgllee}@etri.re.kr

Abstract ¾ **Smartphone is becoming more and more popular in mobile market but applications running on smartphone are much fewer than on existing mobile phones. In order to overcome the shortage of smartphone applications, we have developed the sharable mobile platform that can be used on both smartphones and legacy mobile phones. This approach enables many applications for existing mobile phone to run on smartphone without any changes. This paper introduces the Korean standard mobile platform, Wireless Internet Platform for Interoperability (WIPI) and its architecture. We embody WIPI reference implementation for embedded linux and incorporate it into the linux-powered reference board.**

Keywords ¾ **WIPI, Linux, Smartphone, Qplus**

1. Introduction

1.1 WIPI overview

Korea had the first successful commercialized CDMA mobile communications system and CDMA (3GPP2) is one of the two main global standards for communications technologies along with WCDMA. Korean handset manufacturers are also making great progress in the global market. Samsung Electronics is currently ranked third and LG Electronics is ranked fifth for mobile handset production worldwide [1]. Korean telecommunication industries are attempting to make a standard mobile platform called WIPI, which is the common platform for running mobile applications independent of service provider or handset vendor. Many contents for WIPI have been developed and commercialized during 2003.

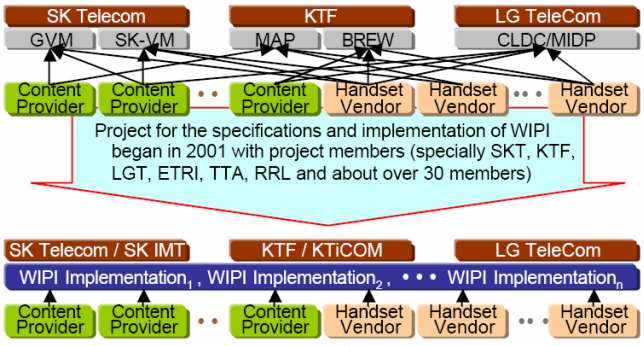


Figure 1. Motivation of emerging WIPI

Figure 1 describes the motivation for the development of WIPI. There are three big mobile carriers in Korea; SK Telecom, KTF and LG Telecom. Before the emergence of WIPI, KTF had adopted BREW as its mobile platform, while the other two carriers SK Telecom and LG Telecom used independent wireless technology based on J2ME. Because the three mobile operators have different technologies, content providers had to pay additional development costs due to modification which occurred several times to make the contents suitable for each mobile service provider. In addition, the availability of multiple platforms was one of the hindering factors that allowed proprietary wireless networks and suppressed freedom of content usage.

A common mobile platform benefits carriers, handset vendors, content developers and customers. The benefits towards each sector are highlighted below:

Carriers: fast delivery of new applications and services, more downloadable services over-the-air(OTA)

Handset vendors: reduction in engineering cost and time, easy working with 3rd party developers

Content developers: open standard mobile platform for developing high quality contents, wider distribution channel and wider array of content services

Customers: choice of various contents independently for wireless carriers and handset vendors

The latest version of WIPI is 2.0.1, released by the Korea Wireless Internet Standardization Forum (KWISF) that consists of about 30 Korean telecommunication industries. Telecommunication Technology Association (TTA) in Korea adopted WIPI as a national mobile standard platform, which means all mobile handsets must support WIPI. In the near future, with WIPI technology-enabled handsets, consumers will be able to personalize their handset with applications such as games, infotainment, and location-based services. The advent and development of WIPI will pave the way for Korean telecom companies to lead the global wireless Internet industry.

1.2 Future of linux-based smartphone

The 3G mobile communication system will be a significant step forward in the convergence of telecommunication and data communication industries. Enhancing the smartphone to merge multiple functions such as voice, data, internet and multi media services will meet these requirements.

Adoption of full-feature handsets – mobile terminals based on full-feature operating systems such as Palm, Linux or WinCE - will represent the next stage of technology evolution in the mobile handset market, as shown in Figure 2. These devices will provide significantly greater design flexibility to OEMs, ease the process of launching new mobile interactive services, and put new advanced computing capabilities in the pockets of consumers [3].

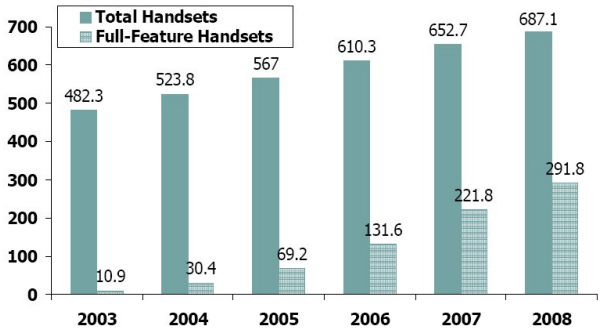


Figure 2. Full-feature handset market

Linux may become the preferred operating system in full-featured mobile terminals as well as a variety of embedded systems. The world of telecommunications, electronics and service industries are closely watching whether linux will dominate the smartphone market and the development of the linux-based smartphone for commercialization. For examples, Motorola announced its first embedded linux-based smartphone, A760, earlier last year. NTT DoCoMo, has adopted Linux for its 3G phone. Samsung is shipping a smartphone powered by embedded Linux from Mizi Research. The WIPI mobile teminal, SCH-i519, is being distributed in China currently.

Table 1 shows that Linux will dominate the smartphone market, beating out rival operating systems. Zelos[3] says that Linux scored the highest on the two criteria that matter most to OEMs and carriers: open-ness and low cost.

Table 1. Long-Term success scores for mobile platforms

Platform	Business Viability (2)	Complet-ness (2)	Cost (3)	End-User Appleal (1)	Openness (3)	Weighted Score
Linux	4	2	5	1	5	43
Palm	2	4	2	4	3	31
Symbian	3	5	3	1	4	38
Windows	5	4	2	4	2	34

Linux-powered smartphone will hold an important position in the near future, so rich applications as well as new mobile platforms will be required. It may be an ideal case that existing content already developed for legacy 2G or 3G mobile phones are executable on smartphone without any changes. The remainder of this paper investigates the existing mobile platform, the WIPI, and presents our approach to adopt the WIPI platform on smartphone.

2. Analysis and Design

2.1 WIPI architecture for handset

There are worldwide platforms such as J2ME of SUN, BREW of QUALCOMM, Symbian etc. Comparing with other platforms WIPI has several key advantages as the handset platform of choice for interoperability.

WIPI supports multiple programming languages such as C/C++ and Java, and downloads and runs all applications as a type of binary from the contents distribution server. Although the application is written in java language, ahead-of-time compiler (AOTC) converts byte code compiled by java compiler (javac) into machine code that can be directly executed on handset.

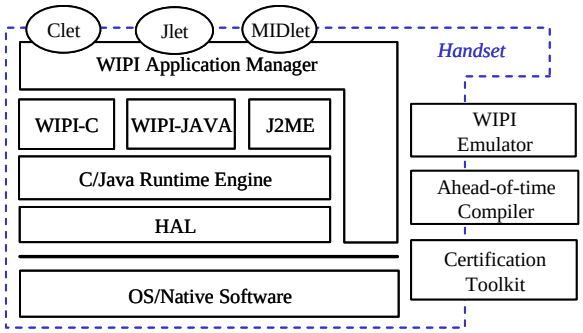


Figure 3. General WIPI architecture

Figure3 shows the general WIPI architecture and additional WIPI-related tools. The lowest layer consists of hardware, OS, and native software. For CDMA enabled-handset, Qualcomm's REX/DMSS is used. For smartphone, native system software is based on WinCE, Linux, Symbian, PalmOS, etc.

The handset adaptation layer (HAL) provides high portability for the above layers such as runtime engine, WIPI-C, WIPI-JAVA and WIPI applications. HAL hides the complex operation of the underlying hardware devices and simplifies the access and control of them.

Runtime engine provides the execution environment for WIPI applications like java virtual machine (JVM) in Java world. The main difference is that JVM loads and executes java class files while WIPI engine takes and executes platform-specific machine code. Runtime engine consists of linker and loader, memory manager, event handler, garbage collector, thread manager, synchronization manager, exception handler and runtime library for supporting the converted code.

API layer provides rich libraries for developing WIPI applications (Clet, Jlet, or MIDlet). Clet is a program written in WIPI-C and Jlet is written in WIPI-JAVA, and MIDlet is a small application written in CLDC/MIDP of J2ME.

WIPI Application Manager (WAM) installs, deletes, lists, or searches applications. WAM can be also written using WIPI-C or WIPI-JAVA. This means that it can be easily ported to any target including WIPI platform.

2.2 WIPI-related technology

The previous section explained the WIPI software architecture installed on the mobile terminal. But there are additional tools that help developers test and emulate WIPI applications, or verify and certify the platform on specific hardware.

AOTC internally consists of 2 main parts, a Java-to-C translator and a cross-compiler. Java-to-C translator converts java class files into C source code, and then cross-compiler converts it into executable binary image. The implementation of WIPI-JAVA APIs includes about 300 native functions which are used for performance improvement or direct access to hardware resources. Because the native code is dependant

on the underlying operating system (OS), it needs to be rewritten for each OS platform. These native functions written in C must be linked with C code translated by Java-to-C translator. The details are beyond the scope of this overview. There are several research papers on AOTC technologies such as GCJ[4], Caffeine[5] and Toba[6].

WIPI emulator provides an integrated development environment (IDE) for WIPI content developers who may have difficulty in accessing the handset platform. Once WIPI application runs without any problem on WIPI emulator, it insures that the application is interoperable with WIPI platform on any device.

To check the interoperability of WIPI implementation from different company, each platform is tested, verified or certified by the platform certification toolkit (PCT) and HAL certification toolkit (HCT)

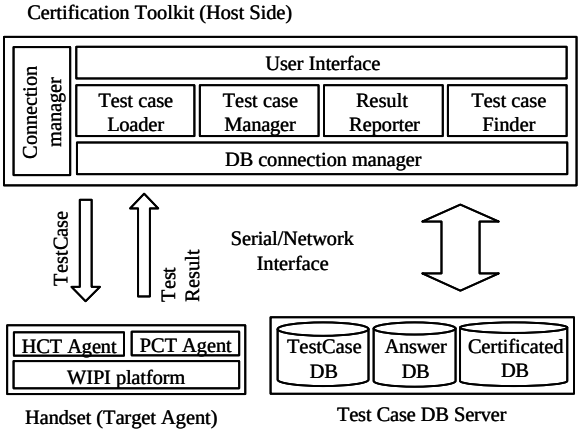


Figure 4. PCT overview

PCT is a suite of tests, tools that determines whether or not implemented WIPI APIs (WIPI-C/WIPI-JAVA) comply with WIPI specification. As shown in Figure 4, PCT consists of 3 different parts; PCT tool on host, database server for saving test suites and PCT agent on mobile terminal. These parts are developed and managed by specific companies that are authorized by KWISF memberships. Database sever contains sharable test case suites for verifying the platform. The tool is a GUI-based tool that communicates with PCT agent. PCT agent program is a Clet or Jlet, so it is downloadable through distributed channel in wireless environment and executable on WIPI. To check the interoperability of the platform, PCT server sends test query to PCT agent. PCT agent receives it and executes server's order by using implemented WIPI APIs in mobile terminal, then sends the results to server. Finally, server determines whether or not the corresponding platform complies with standard specification, by comparing the result to the value stored in its database.

HCT is a suite of tests, tools that checks whether or not implemented HAL APIs comply with WIPI specification. It is mainly used by mobile terminal vendors to test whether the HAL layer is correctly ported to their product. HCT also consists of sever side and client side like a PCT. Because HAL is ported to vendor-specific software at first step, HCT agent has to be developed using native software, not WIPI APIs. This means that HCT agent needs to be rewritten for every target platform. The process of verifying the platform is very similar to PCT.

Until now, there has been no study on HCT, while commercial-quality PCT was already developed by EXE-MOBILE in 2003. We implement HAL APIs for embedded linux and verify it by using HCT that we designed and developed. This work will be the starting point of WIPI reference implementation for linux-based smartphone.

3. WIPI platform for linux-based smartphone

WIPI has been only ported to CDMA-enabled handset and WINDOWS-based PC for emulating the operation of WIPI at current work, in spite of the trends in increasing the number of linux-based smartphone or linux-powered PDA.

The aim of our work is to implement the middleware platform for embedded linux-based smartphone complying with WIPI specification. It has important meaning that existing WIPI contents and tools for supporting WIPI – already developed Clet, Jlet, MIDlet, AOTC, PCT, HCT, contents distribution server, etc. - can be reusable without any modification or with the smallest change.

This section designs and implements the prototype of smartphone software architecture for supporting WIPI.

3.1 Smartphone architecture

Figure 5 proposes the embedded linux-based smartphone software stack containing WIPI. HAL is implemented with Qplus[11] and linux-based libraries, instead of REX/DMSS in CDMA-enabled handset. Qplus is a highly configurable embedded linux system developed by ETRI in Korea. Qplus supports various embedded systems such as HomeServer of ETRI, iPAQ of COMPAQ, Zaurus of SHARP, S3C24X0 of SAMSUNG, etc.

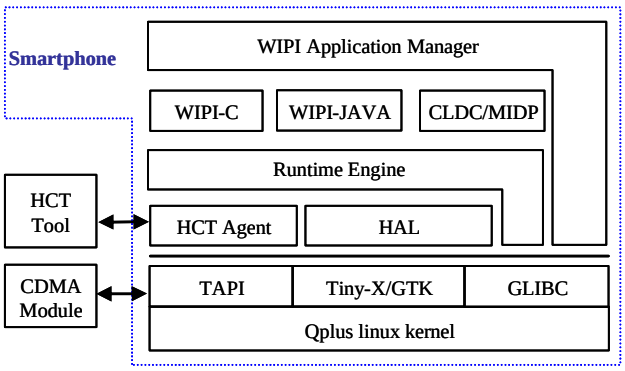


Figure 5. Software stack for linux-powered smartphone

TAPI provides telephony APIs to handle internal or external CDMA module. Tiny-X/GTK is used for creating GUI on LCD and GLIBC is used as the standard C library and Matchbox[12] is used as a basic browser for demonstration.

HAL is implemented using native software such as Glibc, TAPI, Tiny-X/GTK, etc. HCT agent is also a kind of native linux-based application that can communicate with HCT tool to verify the HAL implementation on linux-based smartphone.

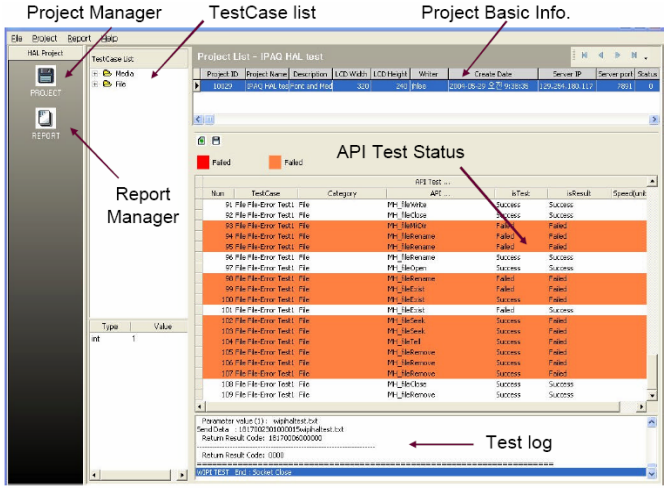
Runtime engine is also dependant on underlying layer and

it must be developed with linux-based libraries, while WIPI APIs and WAM are embodied, as much as possible, with HAL APIs to port easily to other platform.

3.2 HAL APIs and HCT

WIPI specification for HAL APIs provides the following functions; System, Call, Handset device, Network, Serial, Media, Time, Utility, File, Input method, Font, Frame buffer, Virtual key. The details are too much space to explain here and they can be referenced in the latest WIPI specifications from official site [9].

HCT tool is an easy-to-use GUI-based tool for testing, verifying, certifying the HAL implementation for each platform. The tool consists of project manager and report manager as shown in Figure 6. The project manager creates new project information and modifies or deletes it from menu. During project creation, in order to determine HAL APIs are to be tested, test cases list should be configured from pre-defined test suites list in HCT database. The tool sends the user-defined test cases to the HCT agent. The agent executes the test cases on iPAQ and returns the result to the tool. While communicating with agent, report manager shows the processing status of HAL APIs. After completion of test, the summary of results is saved in the form of HTML file to view easily.



Fingure 6. HCT on host

3.3 WIPI-C APIs and runtime engine

WIPI specification for WPI-C APIs provides the following functions; Kernel, Graphics, File, Database, Network, Serial communication, UI component, Utility, Generic IO, Terminal Resource, Media Handler, Telephony(Call), SMS, Location information(GPS), Security Communication, Supplementary devices control, Mathematical operation, and Standard C library. The details are too much space to explain here and they can be referenced in the latest WIPI specifications from official site [9].

We implements most of WIPI-C APIs with only HAL APIs to ease port this layer to other platform using non-linux operating system. To verify WIPI-C APIs implementation, we use PCT. The results will be shown in next section.

Runtime engine consists of Dynamic Link Library (DLL) manager, linker and loader, thread manager, memory manager, scheduler and event hander as shown in Figure 7.

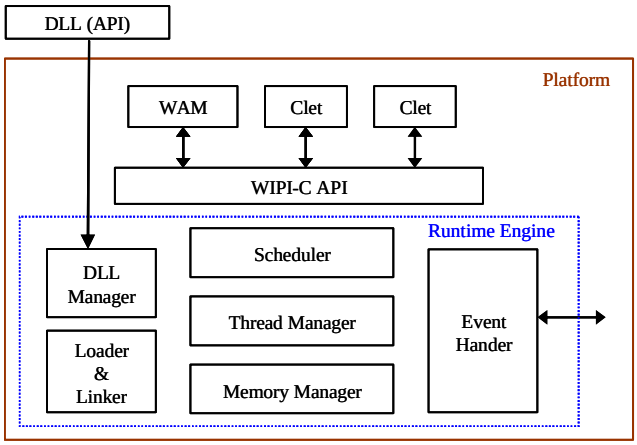


Figure 7. Runtime engine structure

DLL manager adds new API and updates existing API in platform through wireless distribution channel as necessary, without re-compiling built-in programs. Loader reads Clet from embedded file system (EFS) to executable memory area and linker interprets symbols in Clet image. Thread manager provides synchronization mechanism for protecting shared resource and supports multi-threading for running multiple Clets simultaneously. Scheduler uses round-robin algorithm to do context switch. Memory manager allocates or releases storage to and from big memory as needed, and supports memory compaction. Event handler processes various events between WIPI applications and platform. There are two event types (application event, platform event). These events are well defined in WIPI specification. We implement runtime engine with ANSI-C and HAL APIs as much as possible to have high portability.

4. Demonstration

4.1 HAL APIs verification

This section incorporates WIPI reference implementation into linux-based target boards and demonstrates operation of WIPI applications on them.

HAL is deeply dependent on the underlying operating system and physical device. Especially, it is very difficult to develop HAL for the latest original equipment manufacturer (OEM) devices because most of vendors do not open hardware specification or device driver code. This difficulty makes linux more attractive, due to its openness. So, we select iPAQ of COMPAQ, S3C24x0 of Samsung, YOPY-3700 of GMATE, which are easy to update kernel and add custom functionality as required by developers.

Comparing with iPAQ, handsets have different device features such as small size LCD, telephony interface, etc. vibrator and telephony operation in handset are replaced by icons in iPAQ. Display window size is also reduced to

160x240 for the existing WIPI contents developed for phone are as shown in Figure 6. The overview of operation between HCT tool on PC and HCT agent on iPAQ is denoted in Figure 6 and Figure 8.

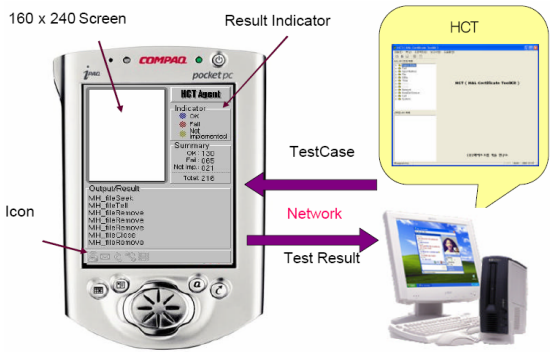


Figure 8. HCT agent on iPAQ

4.2 WIPI-C APIs and runtime engine verification

The table below shows the results from passing WIPI-C APIs implementation through PCT to check whether or not each API is compliant to WIPI specification. Most of APIs pass successfully except 9 cases, shown as Table 2. These APIs are not critical in executing Clet, main reasons are either PCT has faults itself or not-implemented APIs.

Table 2. Result from passing WIPI-C APIs through PCT

	Pass	Fail	Error	Not Run	Total
Kernel	72	1	-	1	74
Graphic Batch	41	-	-	1	42
Graphic Interactive	46	-	-	1	47
Database	50	1	-	-	51
File System	36	-	-	-	36
Media	8	-	-	2	10
Serial	7	-	-	-	7
Telephony	1	-	-	-	1
Utility Batch	7	-	-	-	7
Utility Interactive	3	-	-	-	3
UIC Batch	41	-	-	-	41
UIC Interactive	16	-	-	1	17

We get Clets already developed from WIPI emulator and directly execute them on S3C2440A-based KINGFISH and YOPY-3700 without any change, except image size. Figure 9 demonstrates that existing WIPI applications for handset can be executed without any modifications to the application itself, using our implementation.

Figure 9. Running Clet on linux-based target

5. Conclusion and future work

This paper introduces the mobile trends in Korea and explains the Korean standard mobile platform called WIPI, which is aimed at providing contents interoperability among mobile operators. Now, WIPI technology has already been formalized in commercial mobile terminals as national standard in Korea. Many WIPI applications have also been developed using the WIPI emulator.

This paper demonstrates the current WIPI platform and expands the embedded linux area to enable existing mobile contents to be executed on linux-powered smartphone without any modification. It means WIPI platform has high possibility to be applied as new mobile platform for another mobile OS such as Symbian, Palm, Nucleus as well as linux or REX.

Currently we have developed some part of the WIPI reference implementation for linux-based smartphone. But we plan to implement WIPI-JAVA API using AOTC and java runtime engine by the first half of 2005, and then add more test suites to PCT or HCT. In AOTC technology, to run Java-based WIPI application, research topics such as reducing the converted code size and runtime library size are bleeding edge research areas. We are pursuing advanced research in reducing the size of the code converted by AOTC in resource-limited mobile terminals.

REFERENCES

[1] Strategy Analytics, “Global Handset Market”, market report, http://www.strategyanalytics.com
[2] http://www.linuxdevices.com.
[3] Seamus McAteer, “Defining the market for fullfeatured handset”, Zelos Group, Inc.
[4] “Guide to GNU gcj”, http://gcc.gnu.org/java/
[5] Cheng-Hsueh A.Hsieh, John C.Gyllenhaal, and Wen-mei W.Hwu, “Java Bytecode to Native code Translation: The Caffeine Prototype and Preliminary Results” Proceedings of the 29th International Symposium on Microarchitecuture, December 1996.
[6] T.A.Proebsting, G.Townsend, P.Bridges, J.H.Hartman, T.Newsham, and S.A. Watterson, “Toba: Java for Applications – A Way Ahead of Time(WAT) Compiler”, Proceeding of the Third Conference on Object-Oriented Technologies and Systems(COOTS), USENIX Association Press, 4153; 1997
[7] KWISF, WIPI 1.2 specification, www.wipi.or.kr
[8] KWISF, WIPI 2.0 specification, www.wipi.or .kr
[9] KWISF, WIPI 2.0.1 specification, www.wipi.or .kr
[10] WIPI developer site, www.developer.wipi.or.kr
[11] Embedded S/W Research Division, ETRI, QPlus, http://qplus.etri
[12] Matchbox project, http://matchbox.handhelds.org